A Framework for Automatically Enforcing Privacy Policies

Jean Yang

MSRC / October 15, 2013
Privacy matters. People get it wrong.

Facebook quickly fixes privacy leak in new timeline, group says

The leak allowed 'friends of friends' to see events another person attended

By Jeremy Kirk
March 17, 2013 10:53 PM ET   1 Comment
Many possible points of failure.

Desired Policy

Only friends can see GPS location.

Implementation

- Policy
  - getLocation(user)
- Policy
  - findAllUsers(location)
- Policy
  - findTopLocations()
Increasingly complex policies.

Desired Policy

Only friends who are local within next five hours can see GPS location.
Developer education would solve this problem. **STRONG REJECT**
Easier if we separate policies from other functionality.

Policy Implementation

Only friends can see GPS location.

Other Implementation

getLocation(user)

findAllUsers(location)

findTopLocations()

Involves integrating with the language semantics.
I CAN HAS PROGRAMMING LANGUAGE?
The Jeeves Language

\[ \langle \text{findAllUsers} \rangle_k \]

Associated with policies.

You have no friends in this location.
Sensitive Values

```scala
val a { low, high }
val loc = < gpsCoords | country(gpsCoords) >
```

Policies

```scala
restrict a: loc.(isNear(oc, jean))
```

Core Functionality

```scala
val msg: String = "Jean’s location is " + asStr(loc)
```

Contextual Enforcement

```scala
print {andrey} msg "Jean’s location is N 52.19°, W 0.13°."
print {rishabh} msg "Jean’s location is in the United Kingdom."
```
Jeeves Policies

\[ \text{restrict } a : \lambda oc. \text{isNear}(oc, me) \]
\[ \lambda oc. \text{addConstraint} (\neg \text{isNear}(oc, me) \implies a == \text{low}) \]

Policies may refer to sensitive values.

\textit{Default values} yield guarantees of maximal functionality.

Constraints imply low.
Always a consistent assignment.
Jeeves Execution

Policy manipulation

label a
restrict a: loc.true

Policies
a → loc.true

Constraints
true ⇒ a = low

Facetted execution

= 3
⟨ 3 | 0 ⟩_a

⟨ true | false ⟩_a

print {…} ...

false
Where is THE THEOREM in this paper?

STRONG

REJECT
Classical Security

Lattice of access levels.

Level 3: top secret.

Level 2: highly classified.

Level 1: privileged information.
Classical Security

Viewers must have access for the highest level.
Jeeves Security
Jeeves Non-Interference Guarantee

Given a sensitive value $\langle L / H \rangle_a$ all executions where $a$ must be low produce equivalent outputs no matter the value of $H$.

Takes into account when label depends on sensitive values!
Non-Interference in Jeeves

\begin{verbatim}
val location = < default | actual >_a
restrict a: λoc.(distance(oc, location) < 25)
\end{verbatim}

Viewer within radius: 
a is allowed to be high.
Non-Interference in Jeeves

```plaintext
val location = < default | actual >\textsubscript{a}
restrict a: \lambda oc. (distance(oc, location) < 25)
```

Viewer outside radius: a must be **low**.
Viewer should not be able to distinguish actual location from any other of these points.
Implementation

Overload operators for faceted evaluation.

Store policies in runtime environment

Use an SMT solver as a model finder.

Jean Yang / Jeeves
Jeeves Frameworks

Scalatra frontend \(\rightarrow\) Scala Embedded DSL \(\rightarrow\) SQL (Squeryl)

Python-Jeeves

source transform

Django frontend \(\rightarrow\) Python Embedded DSL \(\rightarrow\) PostGre SQL
Case study: JConf

JConf policies

JConf functionality

Reviewer

Author
Jconf Architecture

Core Program
- Search papers.
- Display papers.
- Add and remove tags.
- Assign and submit reviews.
## Functionality vs. Policy

<table>
<thead>
<tr>
<th>File</th>
<th>Total LOC</th>
<th>Policy LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConfUser.scala</td>
<td>212</td>
<td>21</td>
</tr>
<tr>
<td>PaperRecord.scala</td>
<td>304</td>
<td>75</td>
</tr>
<tr>
<td>PaperReview.scala</td>
<td>116</td>
<td>32</td>
</tr>
<tr>
<td>ConfContext.scala</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>Backend + Squeryl</td>
<td>800</td>
<td>0</td>
</tr>
<tr>
<td>Frontend (Scalatra)</td>
<td>629</td>
<td>0</td>
</tr>
<tr>
<td>Frontend (SSP)</td>
<td>798</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>2865</strong></td>
<td><strong>&lt; 5% 128</strong></td>
</tr>
</tbody>
</table>
Current Directions.

Jeeves runtime

Database
What about inputs?
Storing multiple versions?

Reviewer A

New review

Reviews

Reviewer B

Old review

New review

Author
When viewers specify trust...

Need to additionally track who is writing...

User → Scores → Old score → New score → User

User → Scores → Old score → New score → User

User → Scores → Old score → New score → User

User → Scores → Old score → New score → User

User → Scores → Old score → New score → User
Low-level mechanism

<table>
<thead>
<tr>
<th>Mutable State</th>
</tr>
</thead>
<tbody>
<tr>
<td>42</td>
</tr>
<tr>
<td>42</td>
</tr>
<tr>
<td>42</td>
</tr>
</tbody>
</table>

\[ p_1, p_2, p_2 \]

Associate references with policies

\textbf{wrestrict} \ \lambda \text{in.} \lambda \text{out.} (\text{isFriends(in, alice)})

\textbf{wrestrict} \ \lambda \text{in.} \lambda \text{out.} (\text{(in == alice) \&\& isFriends(out, alice)})
Guarantees

Execution with writer’s inputs

Execution without writer’s inputs

New score
Write Policy Case Studies

Authentication  Conference management  Battleship game

Notes
• Support policies for confidentiality and integrity.
• Policy-agnosticism good for encoding other policies, for instance game rules.
Interfacing with the dB

select * from papers
where author = "Jean Yang"

author_{high}  |  author_{low}  |  policies

\langle p \rangle
FINALLY.. I CAN FOCUS ON FUNCTIONALITY!
Jeeves Team

Armando Solar-Lezama

Travis Hance

Cormac Flanagan

Thomas Austin

Benjamin Shaibu

CSAIL
The Jeeves Framework